The CLPZINC modeling language and its long compilation chain to solving

Philippe Morignot
Constraint Satisfaction Problems (CSP)

• A paradigm for solving combinatorial problems
• Finite-domain variables, constraints.
  \[ \forall i \in [1, 30], x_i \in [1, 200], y_i \in [1, 200] \]
  \[ \forall i, j, i < j : x_i + i \leq x_j \lor x_j + j \leq x_i \lor y_i + i \leq y_j \lor y_j + j \leq y_i \]
• Reification: \[ \begin{cases} x_1 + 3 \leq x_2 \text{ satisfied: bool } b = 1 \\ x_1 + 3 \leq x_2 \text{ violated: bool } b = 0 \end{cases} \]
• Search strategy: additional constraints which orient search and propagation of the solver.
  – e.g., labelling: enumerating the values, one by one.
ZINC

- A high-level language for stating a CSP
- Zinc specification and the reduced MiniZinc implementation (NICTA).
- **Example:** Korf.

  ```
  int: n = 30;
  int: range = 200;
  array[1..n] of var 1..range: x;
  array[1..n] of var 1..range: y;
  constraint forall(i in 1..n-1, j in i+1..n)
      (x[i] + i <= x[j] \lor x[j] + i <= x[i] \lor y[i] + i <= y[j] \lor y[j] + j <= y[i]);
  solve satisfy;
  ```

- FlatZinc: A low-level language easily parsed by solvers (e.g., CHOCO, JaCoP, SICTUS). A MiniZinc model is compiled into a FlatZinc one.

  MiniZinc -> FlatZinc -> (solvers)

- `solve satisfy;`        `solve minimize X;`        `solve maximize Y;`
Weakness of ZINC

int: n = 30;
int: range = 200;
array[1..n] of var 1..range: x;
array[1..n] of var 1..range: y;
constraint forall(i in 1..n-1, j in i+1..n)
    (x[i] + i <= x[j]  ∨  x[j] + i <= x[i]  ∨  y[i] + i <= y[j]  ∨  y[j] + j <= y[i]);
solve :: seq_search (
    int_search(x, input_order, indomain_min, complete),
    int_search(y, first_fail, indomain_split, complete)
) satisfy;

• Zinc’s annotations express little only of search strategies ...
CLPZINC

- Reified constraints express disjunctions

\[( \text{C1} ; ( \text{C2} ; \text{C3} ))\]

- CLPZINC is a Horn-clause extension of MiniZinc for expressing search strategies in CLP clauses.
  - e.g., labelling: \textit{between}(A, X, B) :-
    \[A \leq B, \ (X = A \ ; \ S = A + 1, \ \text{between}(S, X, B)).\]

- Extended with records, forall, exists
Example of search strategies

interval_splitting(\(X, \text{Step}, \text{Min}, \text{Max}\)) :-

\[ \text{Min} + \text{Step} \leq \text{Max}, \]
\[ \text{NextX} = \text{min}(X) + \text{Step}, \]
\( (X < \text{NextX} ; \]
\[ X \geq \text{NextX}, \]
\[ \text{interval_splitting}(X, \text{Step}, \text{Min} + \text{Step}, \text{Max}) \). \]

interval_splitting(\(X, \text{Step}, \text{Min}, \text{Max}\)) :-

\[ \text{Min} + \text{Step} > \text{Max}. \]
Packing

- Optimal packing of objects with complex shapes
- Allen, RCC, PKML libraries

- Applications:
  - Korf: How to place $n$ non-overlapping squares, of size $1 \times 1$ up to $nxn$, in a minimal surface?
  - Real data from the company PSA
Compilation chain

Korf → PSA

PKML

CLPZINC

MINIZINC

FLATZINC

CHOCO + IBEX

Korf on \( \mathbb{R} \)

PSA on \( \mathbb{R} \)
Packing Knowledge Modelling Language (PKML)

Box

Shifted Box

Shape

Object

or
Allen’s relations between time intervals

<table>
<thead>
<tr>
<th>Relation</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precedes</td>
<td>$precedes(T1, T2, D) :- \ T1.end[D] &lt; T2.start[D]$.</td>
</tr>
<tr>
<td>Meets</td>
<td>$meets(T1, T2, D) :- \ T1.end[D] = T2.start[D]$.</td>
</tr>
<tr>
<td>Overlaps</td>
<td>$overlaps(T1, T2, D) :- \ T1.start[D] &lt; T2.start[D] \wedge \ T1.end[D] &lt; T2.end[D] \wedge T2.start[D] &lt; T1.end[D]$.</td>
</tr>
<tr>
<td>Begins</td>
<td>$begins(T1, T2, D) :- \ T1.end[D] &lt; T2.start[D]$.</td>
</tr>
<tr>
<td>Ends</td>
<td>$ends(T1, T2, D) :- \ T1.end[D] = T2.start[D]$.</td>
</tr>
<tr>
<td>IsContainedBy</td>
<td>$iscontainedby(T1, T2, D) :- \ T1.end[D] &lt; T2.start[D]$.</td>
</tr>
<tr>
<td>IsMetBy</td>
<td>$ismetby(T1, T2, D) :- \ T1.end[D] = T2.start[D]$.</td>
</tr>
<tr>
<td>IsPrecededBy</td>
<td>$isprecededBy(T1, T2, D) :- \ T1.end[D] &lt; T2.start[D]$.</td>
</tr>
<tr>
<td>IsOverlappedBy</td>
<td>$isoverlappedby(T1, T2, D) :- \ T1.start[D] &lt; T2.start[D] \wedge \ T1.end[D] &lt; T2.end[D] \wedge T2.start[D] &lt; T1.end[D]$.</td>
</tr>
</tbody>
</table>
Region Connection Calculus (RCC)

\[
\text{task}(\text{Task}, \text{Start}, \text{End}) :- \text{Task} = (\text{start}: \text{Start}, \text{end}: \text{End}).
\]

\[
\text{disjoint}(T1, T2, N) :- \\
\quad \exists (i \in 1..N) \left( \text{precedes}(T1, T2, i) \lor \text{preceded_by}(T1, T2, i) \right).
\]

\[
\text{overlap}(T1, T2, N) :- \\
\quad \forall (i \in 1..N) \left( \text{overlaps}(T1, T2, i) \right).
\]
Experiments: Integers

• Korf in CLPZINC using CHOCO:

• PSA in CLPZINC: 0.263s
  Gravity; Weight stacking; Weight balancing; Stack oversize.
Real numbers

- Native in MiniZinc/FlatZinc and IBEX
- Added to CHOCO parser and to CLPZINC.

- Example in CLPZINC for packing:
  Approximating real variables on a multi-dimensional grid

  \[
  \text{grid}(Y1, Y2, X, N) :- \\
  abs(Y2 - Y1) \leq 1.0 / (2.0 * \text{int2float}(N)), \\
  Y2 = \text{int2float}(X) / \text{int2float}(N).
  \]
Experiments: Reals

- Korf in CLPZINC using CHOCO + IBEX:
  \[ \text{grid} = [5, 6] \]
Conclusion

• CLPZINC is a modeling language for expressing search strategies for CSP
• Allen, RCC and PKML are CLPZINC libraries
• Real numbers are proposed in the compilation chain
• Application to packing without significant performance loss

• Future work (in another life!):
• Port reals to MiniZinc v2.0 and CHOCO v3.2
• What if objects have curved shapes? e.g., circles with PKML real (CMA-ES) ...